

BBBBBBBBBBBBBBB AAAAAAAA DDDDDDDDDDDDD BBBBBBBBBBBBBB LLL KKK KKK
BBBBBBBBBBBBBBB AAAAAAAA DDDDDDDDDDDDD BBBBBBBBBBBBBB LLL KKK KKK
BBBBBBBBBBBBBBB AAAAAAAA DDDDDDDDDDDDD BBBBBBBBBBBBBB LLL KKK KKK

BBB BBB AAA AAA DDD DDD BBB BBB LLL KKK KKK
BBB BBB AAA AAA DDD DDD BBB BBB LLL KKK KKK
BBB BBB AAA AAA DDD DDD BBB BBB LLL KKK KKK
BBB BBB AAA AAA DDD DDD BBB BBB LLL KKK KKK
BBB BBB AAA AAA DDD DDD BBB BBB LLL KKK KKK
BBB BBB AAA AAA DDD DDD BBB BBB LLL KKK KKK
BBB BBB AAA AAA DDD DDD BBB BBB LLL KKK KKK
BBB BBB AAA AAA DDD DDD BBB BBB LLL KKK KKK

BBBBBBBBBBBBBBB AAA AAA DDD DDD BBBBBBBBBBBBBB LLL KKKKKKKKK
BBBBBBBBBBBBBBB AAA AAA DDD DDD BBBBBBBBBBBBBB LLL KKKKKKKKK
BBBBBBBBBBBBBBB AAA AAA DDD DDD BBBBBBBBBBBBBB LLL KKKKKKKKK

BBB BBB AAAAAAA DDD DDD BBB BBB LLL KKK KKK
BBB BBB AAAAAAA DDD DDD BBB BBB LLL KKK KKK
BBB BBB AAAAAAA DDD DDD BBB BBB LLL KKK KKK
BBB BBB AAA AAA DDD DDD BBB BBB LLL KKK KKK
BBB BBB AAA AAA DDD DDD BBB BBB LLL KKK KKK
BBB BBB AAA AAA DDD DDD BBB BBB LLL KKK KKK
BBB BBB AAA AAA DDD DDD BBB BBB LLL KKK KKK

BBBBBBBBBBBBBBB AAA AAA DDDDDDDDDDDDD BBBBBBBBBBBBBB LLLL KKK KKK
BBBBBBBBBBBBBBB AAA AAA DDDDDDDDDDDDD BBBBBBBBBBBBBB LLLL KKK KKK
BBBBBBBBBBBBBBB AAA AAA DDDDDDDDDDDDD BBBBBBBBBBBBBB LLLL KKK KKK

FILEID**SCANFILE

D 15

```
1 0001 0 MODULE SCANFILE {
2 0002 0 IDENT = 'V04-000'
3 0003 0 )
4 0004 1 BEGIN
5
6
7 0007 1 *****
8 0008 1 *
9 0009 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
10 0010 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
11 0011 1 * ALL RIGHTS RESERVED.
12 0012 1 *
13 0013 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
14 0014 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
15 0015 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
16 0016 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
17 0017 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
18 0018 1 * TRANSFERRED.
19 0019 1 *
20 0020 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
21 0021 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
22 0022 1 * CORPORATION.
23 0023 1 *
24 0024 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
25 0025 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
26 0026 1 *
27 0027 1 *
28 0028 1 *****
29 0029 1 ++
30 0030 1 FACILITY:
31 0031 1 DYNAMIC BAD BLOCK UTILITY
32 0032 1 ABSTRACT:
33 0033 1 THIS PROCESS EXAMINES FILES SUSPECTED OF CONTAINING BAD
34 0034 1 DISK BLOCKS. THOSE DISK BLOCKS VERIFIED TO BE BAD ARE ADDED
35 0035 1 TO THE BAD BLOCK FILE. THE OTHERS ARE RETURNED TO THE VOLUME
36 0036 1 FOR REUSE.
37 0037 1 ENVIRONMENT:
38 0038 1 VAX/VMS OPERATING SYSTEM, VERSION 1.0
39 0039 1 AUTHOR:THOMAS G. DOPIRAK , CREATION DATE:5/16/78
40 0040 1
41 0041 1
42 0042 1 MODIFIED BY:
43 0043 1
44 0044 1 V0002 ACG0059 Andrew C. Goldstein, 21-Aug-1979 20:45
45 0045 1 Fix repeated write/read test so it really repeats
46 0046 1
47 0047 1 --
```

```
49      0048 1 | TABLE OF CONTENTS:  
50      0049 1 |  
51      0050 1 |  
52      0051 1 |  
53      0052 1 | FORWARD ROUTINE  
54      0053 1 | SCAN:NOVALUE,  
55      0054 1 | GROUP_BLOCKTEST,  
56      0055 1 | GROUP_RETURN,  
57      0056 1 | CHECK_BADSTATUS,  
58      0057 1 | NORMAL_COMPLETE:NOVALUE,  
59      0058 1 | ERROR_COMPLETE:NOVALUE,  
60      0059 1 | TRUNCATE,  
61      0060 1 | TRUNCATE_BAD,  
62      0061 1 | BLOCKTEST,  
63      0062 1 | POSITION_TO_EOF,  
64      0063 1 | DO_QIOW  
65      0064 1 | ; !  
66      0065 1 |  
67      0066 1 |  
68      0067 1 | INCLUDE FILES:  
69      0068 1 |  
70      0069 1 |  
71      0070 1 | LIBRARY 'SY$LIBRARY:LIB.L32';  
72      0071 1 |  
73      0072 1 | MACROS:  
74      0073 1 |  
75      0074 1 |  
76      0075 1 | MACRO  
77      0076 1 | DIRECTORY ID=FIB[FIB$W_DID]%, !START OF DIRECTORY ID  
78      M 0077 1 | CSTRING[] = (UPLIT-BYTE(%CHARCOUNT(%STRING(%REMAINING)),  
79      0078 1 | %STRING(%REMAINING)) )%  
80      0079 1 | ;  
81      0080 1 |  
82      0081 1 | VARIOUS DEFINITIONS  
83      0082 1 |  
84      0083 1 |  
85      0084 1 | LITERAL  
86      0085 1 | BLOCK_TEST_SIZE=15, !NUMBER OF BLOCKS IN A GROUP  
87      0086 1 | GROUP_SIZE=BLOCK_TEST_SIZE*512. !NUMBER OF BYTES IN A GROUP  
88      0087 1 | TRIALS_TO_SUC=3; !NUMBER OF TIMES A BLOCK MUST BE SUCCESSFULLY BE WRITTEN  
89      0088 1 |  
90      0089 1 |  
91      0090 1 | OWN STORAGE:  
92      0091 1 |  
93      0092 1 |  
94      0093 1 | OWN  
95      0094 1 |  
96      0095 1 | !BLOCKS OF TEST DATA  
97      0096 1 |  
98      0097 1 | DISK_TEXT:VECTOR[GROUP_SIZE,BYTE],  
99      0098 1 | GROUP_TEST_DATA:VECTOR[GROUP_SIZE,BYTE].  
100     0099 1 |  
101     0100 1 | READ_FAIL, !LOGICAL INDICATING IF GROUP FAILED ON READ  
102     0101 1 | TRUNC_BLOCK, !FIRST VBN ACTUALLY REMOVED IN TRUNCATE  
103     0102 1 | BAD_COUNT:INITIAL(0), !COUNT OF BAD BLOCKS FOUND  
104     0103 1 | STARTING_BLOCK, !FIRST VBN IN GROUP  
105     0104 1 | LAST_BLOCK, !LAST BLOCK IN A GROUP
```

```
106      0105 1  IOSB:VECTOR[4 WORD] INITIAL(0.0),    !IOSB FOR DISK OPERATIONS
107      0106 1  FIB:BLOCK[FIB$C_LENGTH, BYTE]      !FILE IDENTIFICATION BLOCK
108      0107 1  ;
109      0108 1  ;
110      0109 1  ! EQUATED SYMBOLS:
111      0110 1  ;
112      0111 1  ;
113      0112 1  BIND
114      0113 1  ;
115      0114 1  ;
116      0115 1  ! SYMBOLS FOR TYPES OF BLOCK TEST RESULTS
117      0116 1  ;
118      0117 1  NORMAL_STS=0,                      !TEST COMPLETED NORMALLY
119      0118 1  ERROR_STS=1,                       !NON-RECOVERABLE ERROR
120      0119 1  BAD_STS=2,                         !BAD BLOCK INDICATED
121      0120 1  TRUE=1,
122      0121 1  FALSE=0,
123      0122 1  FIB_DESC=UPLIT(FIB$C_LENGTH,FIB)
124      0123 1  ;
125      0124 1  ;
126      0125 1  ! EXTERNAL REFERENCES:
127      0126 1  ;
128      0127 1  ;
129      0128 1  EXTERNAL
130      0129 1  CHANNEL:WORD,
131      0130 1  MBX_CHANNEL:WORD,                  !CHANNEL TO F11ACP MAILBOX
132      0131 1  ACP_MAIL:BLOCK[,BYTE],            !BUFFER FROM F11ACP
133      0132 1  OLD_UCB
134      0133 1  ;
135      0134 1  ;
136      0135 1  EXTERNAL ROUTINE
137      0136 1  SET_UCB;
```

```
139      0137 1 GLOBAL ROUTINE SCAN:NOVALUE=
140      0138 1
141      0139 1 !++
142      0140 1 !FUNCTIONAL DESCRIPTION:
143      0141 1
144      0142 1 MAIN ROUTINE FOR FILE PROCESSING. CONTROLS THE
145      0143 1 EXAMINATION OF THE FILE IN GROUPS AND THE RETURN
146      0144 1 OF THE FILES BLOCKS.
147      0145 1
148      0146 1 !FORMAL PARAMETERS:
149      0147 1
150      0148 1     NONE
151      0149 1
152      0150 1 !IMPLICIT INPUTS:
153      0151 1
154      0152 1     CHANNEL: CHANNEL TO SUSPECT DEVICE
155      0153 1     ACP_MAIL: MAIL FROM F11ACP
156      0154 1
157      0155 1 !IMPLICIT OUTPUTS:
158      0156 1     NONE
159      0157 1
160      0158 1
161      0159 1 !ROUTINE VALUE:
162      0160 1 !COMPLETION CODES:
163      0161 1     NONE
164      0162 1
165      0163 1
166      0164 1 !SIDE EFFECTS:
167      0165 1
168      0166 1     THE SUSPECT FILE IS RETURNED TO THE SYSTEM, BLOCKWISE.
169      0167 1
170      0168 1
171      0169 1
172      0170 2 !-- BEGIN
173      0171 2
174      0172 2 !++ CLEAR THE FIB
175      0173 2
176      0174 2
177      0175 2
178      0176 2     CH$FILL(0,FIBSC_LENGTH,FIB);
179      0177 2
180      0178 2
181      0179 2 !INITIALIZE ACCESS TO A FILE AND INITIALIZE LAST_BLOCK
182      0180 2
183      0181 2     IF
184      0182 2     NOT POSITION_TO_EOF()
185      0183 2     THEN
186      0184 2     RETURN;
187      0185 2
188      0186 2
189      0187 2 !LOOP THROUGH ALL GROUPS IN THE FILE
190      0188 2
191      0189 2     WHILE TRUE DO
192      0190 2     BEGIN
193      0191 2
194      0192 2
195      0193 3     !FIND START OF GROUP TO TEST
```

```
196      0194
197      0195
198      0196
199      0197
200      0198
201      0199
202      0200
203      0201
204      0202
205      0203
206      0204
207      0205
208      0206
209      0207
210      0208
211      0209
212      0210
213      0211
214      0212
215      0213
216      0214
217      0215
218      0216
219      0217
220      0218
221      0219
222      0220
223      0221
224      0222
225      0223
226      0224
227      0225
228      0226
229      0227
230      0228
231      0229
232      0230
233      0231
234      0232
235      0233
236      0234
237      0235
238      0236
239      0237
240      0238
241      0239
242      0240
243      0241
244      0242
245      0243
246      0244
247      0245
248      0246
249      0247
250      0248
251      0249

      IF .LAST_BLOCK LSSU BLOCK_TEST_SIZE
      THEN STARTING_BLOCK=1
      ELSE STARTING_BLOCK=.LAST_BLOCK-BLOCK_TEST_SIZE+1;
      !*
      !TEST GROUP OF BLOCKS
      !ACTION DEPENDS UPON WHETHER ANY 'BAD' BLOCKS FOUND
      CASE CHECK_BADSTATUS(GROUP_BLOCKTEST()) FROM NORMAL_STS TO BAD_STS OF
      SET [NORMAL_STS]:           !SUCCESSFUL TEST
      IF++                         !CHECK TO SEE IF FINISHED
      THEN .STARTING_BLOCK EQLU 1
      BEGIN
      NORMAL_COMPLETE();
      RETURN
      END
      ELSE
      LAST_BLOCK=.STARTING_BLOCK-1;
      [ERROR_STS]:                 !ERROR BUT NOT BADBLOCK ERROR
      BEGIN
      ERROR_COMPLETE();
      RETURN
      END;
      [BAD_STS]:                  !BADBLOCK FOUND, SCAN INDIVIDUAL BLOCKS
      BEGIN
      IF NOT GROUP_RETURN()
      THEN
      BEGIN
      ERROR_COMPLETE();
      RETURN
      END;
      IF (.STARTING_BLOCK EQLU 1) OR
      (.TRUNC_BLOCK LEQ 1)
      THEN
      BEGIN
      NORMAL_COMPLETE();
      RETURN
      END
      ELSE
      LAST_BLOCK=.TRUNC_BLOCK-1
      END;
      TES:
      END;
      END;
```

.TITLE SCANFILE
.IDENT \V04-000\

.PSECT SPLITS,NOWRT,NOEXE,2

00000040 00000 P.AAA: .LONG 64
00000000 00004 .ADDRESS FIB

.PSECT SOWNS,NOEXE,2

00000 DISK_TEXT:

.BLKB 7680

01E00 GROUP_TEST DATA:

.BLKB 7680

03C00 READ_FAIL:

.BLKB 4

03C04 TRUNC_BLOCK:

.BLKB 4

00000000 03C08 BAD_COUNT:

.LONG 0

03C0C STARTING_BLOCK:

.BLKB 4

03C10 LAST_BLOCK:

.BLKB 4

00000000 00000000 03C14 IOSB: .LONG 0

03C1C FIB: .BLKB 64

NORMAL_STS= 0

ERROR_STS= 1

BAD_STS= 2

TRUE= 1

FALSE= 0

FIB_DESC= P.AAA

.EXTRN CHANNEL, MBX_CHANNEL

.EXTRN ACP_MAIL, OLD_UCB

.EXTRN SET_UCB

.PSECT SCODES,NOWRT,2

0040	8F	00	56	0000	CF	007C	00000	.ENTRY	SCAN, Save R2,R3,R4,R5,R6	0137
			6E	00	00	9E	00002	MOVAB	STARTING_BLOCK, R6	0176
				10	A6	2C	00007	MOVC5	#0, (SP), #0, #64, FIB	
				0000V	CF	00	FB	CALLS	#0, POSITION_TO_EOF	0182
					59	50	E9	BLBC	R0, 11\$	
					OF	04	A6	CMPL	LAST_BLOCK, #15	0196
						04	D1	BGEQU	2\$	
						05	1E	MOVL	#1, STARTING_BLOCK	0198
						66	01	BRB	3\$	
						05	00	SUBL3	#14, LAST_BLOCK, STARTING_BLOCK	0200
						04	C3	CALLS	#0, GROUP_BLOCKTEST	0206
				0000V	CF	00	FB	PUSHL	R0	
						50	00028	CALLS	#1, CHECK_BADSTATUS	
				0000V	CF	01	0002F	CASEL	R0, #0, #2	
					00	50	CF	WORD	5\$-4\$,-	
02		0012	001A			0006	00038	4\$:	7\$-4\$,-	

					6\$-4\$		
					STARTING_BLOCK, #1		0213
					9\$		
04	A6	66	66	D1 0003E 5\$: CMPL BEQL	#1, STARTING_BLOCK, LAST_BLOCK		0220
				20 13 00041	SUBL3		0212
				01 C3 00043	BRB	1\$	0230
		0000V CF		CE 11 00048	CALLS	#0, GROUP_RETURN	0233
		06		00 FB 0004A 6\$: BLBS	R0, 8\$		0232
		0000V CF		50 E8 0004F	CALLS	#0, ERROR_COMPLETE	0237
				00 FB 00052 7\$: RET			0238
				04 00057 8\$: CMPL	STARTING_BLOCK, #1		0241
				06 13 0005B	BEQL	9\$	0240
		01 F8	A6	D1 0005D	CMPL	TRUNC_BLOCK, #1	0245
				06 14 00061	BGTR	10\$	0189
		0000V CF		00 FB 00063 9\$: CALLS	#0, NORMAL_COMPLETE		0249
				04 00068	RET		
04	A6	F8 A6		01 C3 00069 10\$: SUBL3	#1, TRUNC_BLOCK, LAST_BLOCK		
				A7 11 0006F	BRB	1\$	
				04 00071 11\$: RET			

; Routine Size: 114 bytes, Routine Base: \$CODE\$ + 0000

```
254      0250 1 ROUTINE POSITION_TO_EOF=
255      0251 1
256      0252 1 ++
257      0253 1 FUNCTIONAL DESCRIPTION:
258      0254 1
259      0255 1 ROUTINE INITIALIZES THE FIB, ACCESSES THE FILE WHOSE
260      0256 1 FID IS THE ACP_MAIL, AND DETERMINES THE FILES LENGTH
261      0257 1 IN BLOCKS
262      0258 1
263      0259 1 FORMAL PARAMETERS:
264      0260 1
265      0261 1     NONE
266      0262 1
267      0263 1 IMPLICIT INPUTS:
268      0264 1
269      0265 1     ACP_MAIL[BBSSW_FID]: FILE ID OF SUSPECT FILE
270      0266 1
271      0267 1 IMPLICIT OUTPUTS:
272      0268 1
273      0269 1     LAST_BLOCK: TOTAL NUMBER OF BLOCKS IN FILE
274      0270 1     FIB: ASSORTED FIELDS SET BY IOS_ACCESS
275      0271 1
276      0272 1 ROUTINE VALUE:
277      0273 1 COMPLETION CODES:
278      0274 1
279      0275 1     IF IOS_ACCESS FAILS THEN THAT CODE IS RETURNED
280      0276 1
281      0277 1 SIDE EFFECTS:
282      0278 1
283      0279 1     NONE
284      0280 1
285      0281 1 --
286      0282 1
287      0283 2 BEGIN
288      0284 2
289      0285 2
290      0286 2 OWN
291      0287 2     STAT_BLOCK:VECTOR[5,WORD],           !SPACE FOR FILE STATISTICS BLOCK
292      0288 2                           !RETURNED BY IOS_ACCESS
293      0289 2     ATTRIBUTES:VECTOR[3]
294      0290 2             INITIAL(ATRSC_STATBLK^16+10,STAT_BLOCK,0)
295      0291 2
296      0292 2
297      0293 2 /*SET FILE ACCESS ATTRIBUTES
298      0294 2
299      0295 2     FIB[FIBSV_WRITE]=1;
300      0296 2     FIB[FIBSV_TRUNC]=1;
301      0297 2
302      0298 2 /*PUSH FILE ID INTO FIB
303      0299 2
304      0300 2
305      0301 2     CH$MOVE(6,ACP_MAIL[BBSSW_FID],FIB[FIBSW_FID]);
306      0302 2
307      0303 2
308      0304 2 /*OPEN THE SPECIFIED FILE AND GET ITS SIZE IN BLOCKS
309      0305 2
310      0306 2     DO_QIOW(IOS_ACCESS+IOSM_ACCESS,FIB_DESC,0,0,0,ATTRIBUTES);
```

```

311    0307 2
312    0308 2
313    0309 2 * MOVE THE WORD SWAPPED VIRTUAL BLOCK NUMBER
314    0310 2
315    0311 2 LAST_BLOCK<0,16>=.STAT_BLOCK[3];
316    0312 2 LAST_BLOCK<16,16>=.STAT_BLOCK[2];
317    0313 2
318    0314 2 RETURN TRUE
319    0315 1 END;

```

.PSECT \$0WNS,NOEXE,2

		03C5C STAT_BLOCK:
		.BLKB 10
	0009000A	03C66 .BLKB 2
		03C68 ATTRIBUTES:
	00000000	.LONG 589834
	00000000	.ADDRESS STAT_BLOCK
	00000000	.LONG 0

.PSECT \$CODE\$,NOWRT,2

007C 00000 POSITION_TO_EOF:											
10	A6	0000G	CF	56	0000	CF	9E	00002	WORD Save R2,R3,R4,R5,R6	0250	
				OD	A6	01	88	00007	MOVAB LAST_BLOCK, R6	0296	
				23	A6	01	88	0000B	BISB2 #1, FIB+1	0297	
						06	28	0000F	BISB2 #1, FIB+23	0301	
						58	A6	00016	MOVC3 #6, ACP MAIL+12, FIB+4	0306	
						7E	7C	00019	PUSHAB ATTRIBUTES		
						7E	D4	0001B	CLRQ -(SP)		
						0000	CF	9F	0001D	CLRL -(SP)	
						72	8F	9A	00021	PUSHAB FIB DESC	
						06	FB	00025	MOVZBL #114, -(SP)		
				0000V	CF	02	A6	52	A6	CALLS #6, DO QIOW	
						66	50	A6	BO	MOVW STAT_BLOCK+6, LAST_BLOCK	0311
						50	01	BO	0002E	MOVW STAT_BLOCK+4, LAST_BLOCK+2	0312
						01	00	00033	MOVL #1, R0	0314	
						04	00	00036	RET	0315	

: Routine Size: 55 bytes, Routine Base: \$CODE\$ + 0072

```
321      0316 1 ROUTINE TRUNCATE(VBN)=
322      0317 1 ++
323      0318 1 FUNCTIONAL DESCRIPTION:
324      0319 1
325      0320 1 ROUTINE TRUNCATES OF THE END OF THE CURRENT FILE
326      0321 1 STARTING AT THE INDICATED BLOCK NUMBER. BECAUSE OF
327      0322 1 CLUSTERING NOT ALL BLOCKS REQUESTED MAY BE TRUNCATED
328      0323 1 .LAST BLOCK TRUNCATED IS PLACED INTO TRUNC_BLOCK.
329      0324 1
330      0325 1 FORMAL PARAMETERS:
331      0326 1
332      0327 1     VBN: VIRTUAL BLOCK AT WHICH TO START TRUNCATE
333      0328 1
334      0329 1 IMPLICIT INPUTS:
335      0330 1     NONE
336      0331 1
337      0332 1 IMPLICIT OUTPUTS:
338      0333 1     NONE
339      0334 1
340      0335 1 ROUTINE VALUE:
341      0336 1 COMPLETION CODES:
342      0337 1     STATUS OF IOS_MODIFY OPERATION IS RETURNED
343      0338 1
344      0339 1 SIDE EFFECTS:
345      0340 1     NONE
346      0341 1
347      0342 1
348      0343 1
349      0344 1
350      0345 1
351      0346 1
352      0347 1
353      0348 1
354      0349 2 BEGIN
355      0350 2 LOCAL
356      0351 2     STATUS;
357      0352 2
358      0353 2
359      0354 2 !SET BLOCK TO TRUNCATE AT
360      0355 2     FIB[FIBSL_EXVBN]=.VBN;
361      0356 2
362      0357 2
363      0358 2
364      0359 2 !TRUNCATE A PIECE OFF OF FILE
365      0360 2     STATUS=DO_QIOW(IOS_MODIFY,FIB_DESC,0,0,0,0);
366      0361 2
367      0362 2 !CLEAR SIZE FIELD
368      0363 2     FIB[FIBSL_EXSZ]=0;
369      0364 2
370      0365 2
371      0366 2
372      0367 2 !CHECK FOR ROUNDING FROM CLUSTERING
373      0368 2     IF
374      0369 2     THEN .VBN NEQ .FIB[FIBSL_EXVBN]
375      0370 2
376      0371 2
377      0372 2
```

```

: 378      0373 2      TRUNC_BLOCK=.FIB[FIB$L_EXVBN]
: 379      0374 2      ELSE
: 380      0375 2      TRUNC_BLOCK=.VBN;
: 381      0376 2
: 382      0377 2      RETURN .STATUS
: 383      0378 2
: 384      0379 1      END;

```

0004 00000 TRUNCATE:

				WORD	Save R2	
52	0000'	CF	9E 00002	MOVAB	FIB+28, R2	: 0316
62	04	AC	D0 00007	MOVL	VBN, FIB+28	: 0356
			7E 7C 0000B	CLRQ	-(SP)	: 0361
			7E 7C 0000D	CLRQ	-(SP)	
	0000'	CF	9F 0000F	PUSHAB	FIB_DESC	
			36 DD 00013	PUSHL	#54	
0000V	CF		06 FB 00015	CALLS	#6, D0_010W	
		FC	A2 D4 0001A	CLRL	FIB+24	: 0365
62	04	AC	D1 0001D	CMPL	VBN, FIB+28	: 0371
			05 13 00021	BEQL	1S	
CC	A2		62 D0 00023	MOVL	FIB+28, TRUNC_BLOCK	: 0373
			04 00027	RET		
CC	A2	04	AC D0 00028	MOVL	VBN, TRUNC_BLOCK	: 0375
			1\$: 04 0002D	RET		: 0379

; Routine Size: 46 bytes, Routine Base: \$CODE\$ + 00A9

```
386    0380 1 ROUTINE TRUNCATE_BAD(VBN)=
387    0381 1
388    0382 1 |++
389    0383 1 |  FUNCTIONAL DESCRIPTION:
390    0384 1
391    0385 1      TRUNCATE BAD PERFORMS 2 TRUNCATION OPERATIONS.
392    0386 1      ALL BLOCKS AFTER(HIGHER VBN'S) ARE RETURNED TO
393    0387 1      THE SYSTEM VIA A CALL TO TRUNCATE. THE CURRENT
394    0388 1      VBN KNOWN AS 'BAD' IS TRUNCATED OFF THE CURRENT FILE
395    0389 1      AND ONTO THE BAD BLOCK FILE. DUE TO CLUSTERING, MORE BLOCKS
396    0390 1      THAN REQUESTED MAY BE ADDED TO THE BAD BLOCK FILE AND
397    0391 1      TRUNC_BLOCK IS SET TO THE LAST BLOCK ADDED.
398    0392 1
399    0393 1      FORMAL PARAMETERS:
400    0394 1
401    0395 1          VBN: VIRTUAL BLOCK NUMBER OF BLOCK TO MARK BAD
402    0396 1
403    0397 1      IMPLICIT INPUTS:
404    0398 1          NONE
405    0399 1
406    0400 1
407    0401 1      IMPLICIT OUTPUTS:
408    0402 1
409    0403 1          TRUNC_BLOCK: LAST BLOCK(LOWEST VBN) ADDED TO BAD BLOCK FILE
410    0404 1
411    0405 1      ROUTINE VALUE:
412    0406 1      COMPLETION CODES:
413    0407 1
414    0408 1          IF EITHER TRUNCATE OPERATION FAILS THEN THAT STATUS IS RETURNED
415    0409 1
416    0410 1      SIDE EFFECTS:
417    0411 1
418    0412 1          NONE
419    0413 1
420    0414 1      !--
421    0415 1
422    0416 2 BEGIN
423    0417 2 LOCAL
424    0418 2 STATUS:
425    0419 2
426    0420 2      BAD_COUNT=.BAD_COUNT+1;
427    0421 2
428    0422 2      !TRUNCATE OFF GOOD PORTIONS OF FILE
429    0423 2
430    0424 2      STATUS=TRUNCATE(.VBN+1);
431    0425 2      IF
432    0426 2          (.STATUS NEQ SSS_NORMAL) AND
433    0427 2          (.STATUS NEQ SSS_ENDOFFILE)
434    0428 2      THEN
435    0429 2          RETURN .STATUS;
436    0430 2
437    0431 2      !SET BLOCK TO TRUNCATE AT
438    0432 2
439    0433 2      FIB[FIBSL_EXVBN]=.VBN;
440    0434 2
441    0435 2
442    0436 2      !NOTE RETURN IS TO BAD BLOCK FILE
```

D 16
15-Sep-1984 23:36:57 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 11:54:33 [BADBLK.SRC]SCANFILE.B32;1

```
443 0437 2 FIB[FIB$V_MARKBAD]=1;
444 0438 2
445 0439 2
446 0440 2
447 0441 2
448 0442 2
449 0443 2
450 0444 2
451 0445 2
452 0446 2
453 0447 2
454 0448 2
455 0449 2
456 0450 2
457 0451 2
458 0452 2
459 0453 2
460 0454 2
461 0455 2
462 0456 2
463 0457 2
464 0458 2
465 0459 2
466 0460 2
467 0461 2
468 0462 2
469 0463 2
470 0464 2
471 0465 2
472 0466 1

    !* TRUNCATE A PIECE OFF OF FILL
    STATUS=DO_QIOW(IOS_MODIFY,FIB_DESC,0,0,0,0);
    !* CLEAR SIZE FIELD
    FIB[FIB$L_EXSZ]=0;

    !* CHECK FOR ROUNDING FROM CLUSTERING
    IF .VBN NEQ .FIB[FIB$L_EXVBN]
    THEN TRUNC_BLOCK=.FIB[FIB$L_EXVBN]
    ELSE TRUNC_BLOCK=.VBN;

    !* CLEAR MARK BAD INDICATOR
    FIB[FIB$V_MARKBAD]=0;
    RETURN .STATUS

END:
```

0004 00000 TRUNCATE_BAD:									
									0380
7E	52	0000'	CF	9E	00002	.WORD	Save R2		
		DD	A2	D6	00007	MOVAB	FIB+28, R2		
	04	AC	01	C1	0000A	INCL	BAD_COUNT		0420
	BF	AF	01	FB	0000F	ADDL3	#1, -VBN, -(SP)		0424
	01		50	D1	00013	CALLS	#1, TRUNCATE		
			09	13	00016	CMPB	STATUS, #1		0426
00000870	8F		50	D1	00018	BEQL	1\$		
			2F	12	0001F	CMPB	STATUS, #2160		0427
	62	04	AC	D0	00021	18:	BNEQ	4\$	
	FB	A2	04	88	00025	MOVL	VBN, FIB+28		0433
			7E	7C	00029	BISB2	#4, FIB+23		0438
			7E	7C	0002B	CLRQ	-(SP)		0443
		0000'	CF	9F	0002D	CLRQ	-(SP)		
			36	DD	00031	PUSHAB	FIB_DESC		
0000V	CF		06	FB	00033	PUSHL	#54		
		FC	A2	D4	00038	CALLS	#6, DO_QIOW		0447
	62	04	AC	D1	0003B	CLRL	FIB+24		0453
			06	13	0003F	CMPB	VBN, FIB+28		
	CC	A2	62	D0	00041	BEQL	2\$		
			05	11	00045	MOVL	FIB+28, TRUNC_BLOCK		0455
						BRB	3\$		

SCANFILE
V04-000

E 16
15-Sep-1984 23:36:57 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 11:54:33 [BADBLK.SRC]SCANFILE.B32;1

Page 14
(7)

CC A2 04 AC D0 00047 2\$: MOVL VBN, TRUNC_BLOCK
FB A2 04 8A 0004C 3\$: BICB2 #4, FIB+23-
 04 00050 4\$: RET

: 0457
: 0463
: 0466

; Routine Size: 81 bytes, Routine Base: \$CODE\$ + 00D7

```
474      0467 1 ROUTINE BLOCKTEST(VBN)=
475      0468 1
476      0469 1 ++
477      0470 1 FUNCTIONAL DESCRIPTION:
478      0471 1
479      0472 1 THIS ROUTINE TESTS A SINGLE VIRTUAL BLOCK
480      0473 1 FOR 'BADNESS'. THE ROUTINE READS THE BLOCK A NUMBER
481      0474 1 OF TIMES, CHECKING FOR A DATA SENSITIVE CONDITION, AND THEN
482      0475 1 WRITES AND READS BACK THE WORST CASE PATTERN. UPON ANY
483      0476 1 ABNORMAL CONDITION THE ROUTINE EXITS WITH THAT STATUS.
484      0477 1
485      0478 1 FORMAL PARAMETERS:
486      0479 1
487      0480 1     VBN: VIRTUAL BLOCK TO BE TESTED
488      0481 1
489      0482 1 IMPLICIT INPUTS:
490      0483 1
491      0484 1     READ_FAIL: A LOGICAL VARIABLE, WHEN TRUE INDICATES THAT
492      0485 1     GROUP BLOCKTEST ENCOUNTERED AN ERROR WHILE READING THE
493      0486 1     USER DATA ON THE CURRENT GROUP. THIS DIRECTS BLOCKTEST
494      0487 1     TO READ THE INDIVIDUAL BLOCKS BEFORE OVERWRITING THEM
495      0488 1
496      0489 1 IMPLICIT OUTPUTS:
497      0490 1
498      0491 1     NONE
499      0492 1
500      0493 1 ROUTINE VALUE:
501      0494 1 COMPLETION CODES:
502      0495 1
503      0496 1     IF ANY QIOW FAILS THEN ITS STATUS IS RETURNED
504      0497 1
505      0498 1 SIDE EFFECTS:
506      0499 1
507      0500 1     NONE
508      0501 1
509      0502 1 --
510      0503 1
511      0504 2 BEGIN
512      0505 2
513      0506 2 LOCAL
514      0507 2     STATUS:
515      0508 2
516      0509 2
517      0510 2 * IF GROUP TEST FAILED IN DATA DEPENDENT MANNER
518      0511 2 !READ THE BLOCK BEFORE OVER WRITING IT
519      0512 2
520      0513 2
521      0514 2 IF .READ_FAIL
522      0515 2 THEN INCRT INDEX FROM 1 TO TRIALS_TO_SUC DO
523      0516 2     IF NOT (STATUS=DO_QIOW(IOS_READVBLK+IOSM_INHRETRY,DISK_TEXT,512,.VBN,0,0))
524      0517 2     THEN RETURN .STATUS;
525      0518 2
526      0519 2
527      0520 2
528      0521 2
529      0522 2 * 2 !BLOCK MUST PASS READ/WRITE TEST MULTIPLE BEFORE BEING MARKED GOOD
```

```

531      0524 2
532      0525
533      0526
534      0527
535      0528
536      0529
537      0530
538      0531
539      0532
540      0533
541      0534
542      0535
543      0536
544      0537
545      0538
546      0539
547      0540
548      0541
549      0542
550      0543
551      0544
552      0545
553      0546
554      0547
555      0548
556      0549
557      0550
558      0551
559      0552
560      0553
561      0554
562      0555 1 END;

      INC TEST_INDEX FROM 1 TO TRIALS_TO_SUC DO
      BEGIN
        !* WRITE TO THE INDICATED DISK BLOCK
        IF NOT(STATUS=DO_QIOW(IOS_WRITEVBLK+IOSM_INHRETRY, GROUP_TEST_DATA, 512, .VBN, 0, 0))
        THEN RETURN .STATUS;
        !* TRY AND READ IT BACK
        IF NOT(STATUS=DO_QIOW(IOS_READVBLK+IOSM_INHRETRY, DISK_TEXT, 512, .VBN, 0, 0))
        THEN RETURN .STATUS;
        !* MAKE SURE ITS THE SAME
        IF CHSNEQ(512, GROUP_TEST_DATA, 512, DISK_TEXT)
        THEN RETURN SSS_PARITY
      END;
      RETURN TRUE

```

00FC 00000 BLOCKTEST:						
						Save R2,R3,R4,R5,R6,R7
57	0000V	CF	9E	00002	MOVAB	DO QIOW, R7
56	0000'	CF	9E	00007	MOVAB	DISK_TEXT, R6
21	3C00	C6	E9	0000C	BLBC	READ FAIL, 28
52		01	DD	00011	MOVL	#1, TEST_INDEX
		7E	7C	00014	1\$:	-(SP)
		04	AC	00016	PUSHL	VBN
7E	0200	8F	3C	00019	MOVZWL	#512, -(SP)
		56	DD	0001E	PUSHL	R6
7E	8031	8F	3C	00020	MOVZWL	#32817, -(SP)
67		06	FB	00025	CALLS	#6, DO QIOW
55		50	DD	00028	MOVL	RO STATUS
3D		55	E9	0002B	BLBC	STATUS, 48
E2		03	F3	0002E	AOBLEQ	#3, TEST_INDEX, 18
52		01	DD	00032	2\$:	#1, TEST_INDEX
54		7E	7C	00035	3\$:	-(SP)
		04	AC	00037	PUSHL	VBN
7E	0200	8F	3C	0003A	MOVZWL	#512, -(SP)
	1E00	C6	9F	0003F	PUSHAB	GROUP_TEST_DATA

0467

0514
05180517
0531

; Routine Size: 135 bytes, Routine Base: SCODES + 0128

```
564      0556 1 ROUTINE GROUP_BLOCKTEST=
565      0557 1
566      0558 1 ++
567      0559 1 FUNCTIONAL DESCRIPTION:
568      0560 1
569      0561 1 ROUTINE TESTS GROUPS OF VIRTUALLY CONTIGUOUS BLOCKS FOR
570      0562 1 'BADNESS'. SHOULD ANY OF THE IO OPERATIONS FAIL
571      0563 1 THE STATUS IS IMMEDIATELY RETURNED. GROUPS ARE READ
572      0564 1 SEVERAL TIMES FOR ERROR.
573      0565 1 A WORST CASE IS WRITTEN TO THE GROUP AND THEN READ BACK.
574      0566 1 THE READ DATA IS COMPARED WITH THAT WRITTEN
575      0567 1
576      0568 1 FORMAL PARAMETERS:
577      0569 1
578      0570 1     NONE
579      0571 1
580      0572 1 IMPLICIT INPUTS:
581      0573 1
582      0574 1     STARTING_BLOCK: FIRST VIRTUAL BLOCK IN GROUP
583      0575 1     LAST_BLOCK: LAST VIRTUAL BLOCK IN GROUP
584      0576 1
585      0577 1 IMPLICIT OUTPUTS:
586      0578 1
587      0579 1     NONE
588      0580 1
589      0581 1 ROUTINE VALUE:
590      0582 1 COMPLETION CODES:
591      0583 1
592      0584 1     NONE
593      0585 1
594      0586 1
595      0587 1
596      0588 1
597      0589 1
598      0590 1
599      0591 1
600      0592 2 BEGIN
601      0593 2
602      0594 2 LOCAL
603      0595 2     CURRENT_SIZE.
604      0596 2     STATUS:
605      0597 2
606      0598 2
607      0599 2
608      0600 2
609      0601 2
610      0602 2
611      0603 2
612      0604 2
613      0605 2
614      0606 2
615      0607 2
616      0608 2
617      0609 2
618      0610 2
619      0611 2
620      0612 2
*
!* FOR SHORT FILES OR FOR THE START OF A FILE, GROUP SIZE MAY BE SHORTER
! THAN THE DEFAULT
*
IF .STARTING_BLOCK EQ 1
THEN CURRENT_SIZE=.LAST_BLOCK*512
ELSE CURRENT_SIZE=GROUP_SIZE;
*
!DEFAULT THAT FAILURES WILL NOT BE DATA SENSITIVE
READ_FAIL=FALSE;
```

```
: 621      0613 2
: 622      0614 2
: 623      0615 2
: 624      0616 2
: 625      0617 2
: 626      0618 2
: 627      0619 2
: 628      0620 2
: 629      0621 2
: 630      0622 2
: 631      0623 2
: 632      0624 2
: 633      0625 2
: 634      0626 2
: 635      0627 2
: 636      0628 2
: 637      0629 2
: 638      0630 2
: 639      0631 2
: 640      0632 2
: 641      0633 2
: 642      0634 2
: 643      0635 2
: 644      0636 2
: 645      0637 2
: 646      0638 2
: 647      0639 2
: 648      0640 2
: 649      0641 2
: 650      0642 2
: 651      0643 2
: 652      0644 2
: 653      0645 2
: 654      0646 2
: 655      0647 2
: 656      0648 2
: 657      0649 2
: 658      0650 2
: 659      0651 2
: 660      0652 2
: 661      0653 2
: 662      0654 2
: 663      0655 2
: 664      0656 2
: 665      0657 2
: 666      0658 2
: 667      0659 2
: 668      0660 1

!* GROUP FAILURE MAY BE DATA SENSITIVE
!READ SEVERAL TIMES BEFORE PASSING TO WRITE/READ TESTING

INCR TEST_INDEX FROM 1 TO TRIALS_TO_SUC DO
  IF NOT (STATUS=DO_QIOW(IOS_READVBLK+IOSM_INHRETRY,DISK_TEXT,.CURRENT_SIZE,.STARTING_BLOCK,0,0))
  THEN BEGIN
    READ FAIL=TRUE;
    RETURN .STATUS
  END;

!* GROUP MUST PASS WRITE/READ TEST MULTIPLE TIMES BEFORE
!BEING CONSIDERED GOOD

INCR TEST_INDEX FROM 1 TO TRIALS_TO_SUC DO
  BEGIN
    !* WRITE TO THE INDICATED DISK BLOCK
    IF NOT(STATUS=DO_QIOW(IOS_WRITEVBLK+IOSM_INHRETRY,GROUP_TEST_DATA,.CURRENT_SIZE,.STARTING_BLOCK
    THEN RETURN .STATUS;

    !* TRY AND READ IT BACK
    IF NOT(STATUS=DO_QIOW(IOS_READVBLK+IOSM_INHRETRY,DISK_TEXT,.CURRENT_SIZE,.STARTING_BLOCK,0,0))
    THEN RETURN .STATUS;

    !* MAKE SURE ITS THE SAME
    IF CH$NEQ(.CURRENT_SIZE, GROUP_TEST_DATA,.CURRENT_SIZE,DISK_TEXT)
    THEN RETURN SSS_PARITY
  END;

  RETURN TRUE
END;
```

01FC 00000 GROUP_BLOCKTEST:
58 0000V CF 9E 00002 .WORD Save R2,R3,R4,R5,R6,R7,R8
57 0000' CF 9E 00007 MOVAB DO_QIOW, R8
MOVAB STARTING_BLOCK, R7

: 0556
:

		01		67	D1 0000C		CMPL	STARTING_BLOCK, #1	: 0604
54	04	A7		07	12 0000F		BNEQ	1\$: 0606
				09	78 00011		ASHL	#9, LAST_BLOCK, CURRENT_SIZE	: 0608
		54	1E00	05	11 00016		BRB	2\$: 0612
			F4	8F	3C 00018	1\$:	MOVZWL	#7680, CURRENT_SIZE	: 0620
		52		A7	D4 0001D	2\$:	CLRL	READ_FAIL	
				01	D0 00020	3\$:	MOVL	#1, TEST_INDEX	
				7E	7C 00023		CLRQ	- (SP)	
				67	DD 00025		PUSHL	STARTING_BLOCK	
				54	DD 00027		PUSHL	CURRENT_SIZE	
				C7	9F 00029		PUSHAB	DISK_TEXT	
		7E	8031	8F	3C 0002D		MOVZWL	#328T7, - (SP)	
		68		06	FB 00032		CALLS	#6, DO_QIOW	
		56		50	D0 00035		MOVL	RO, STATUS	
		06		56	E8 00038		BLBS	STATUS, 4\$	
	F4	A7		01	D0 0003B		MOVL	#1, READ_FAIL	: 0623
				37	11 0003F		BRB	6\$: 0624
DE		52		03	F3 00041	4\$:	AOBLEQ	#3, TEST_INDEX, 3\$: 0619
		55		01	D0 00045		MOVL	#1, TEST_INDEX	: 0630
				7E	7C 00048	5\$:	CLRQ	- (SP)	: 0636
				67	DD 0004A		PUSHL	STARTING_BLOCK	
				54	DD 0004C		PUSHL	CURRENT_SIZE	
		7E	E1F4	C7	9F 0004E		PUSHAB	GROUP_TEST_DATA	
		68	8030	8F	3C 00052		MOVZWL	#32818, - (SP)	
		56		06	FB 00057		CALLS	#6, DO_QIOW	
		18		50	D0 0005A		MOVL	RO, STATUS	
				56	E9 0005D		BLBC	STATUS, 6\$	
				7E	7C 00060		CLRQ	- (SP)	: 0644
				67	DD 00062		PUSHL	STARTING_BLOCK	
				54	DD 00064		PUSHL	CURRENT_SIZE	
		7E	8031	C7	9F 00066		PUSHAB	DISK_TEXT	
		68		8F	3C 0006A		MOVZWL	#328T7, - (SP)	
		56		06	FB 0006F		CALLS	#6, DO_QIOW	
		04		50	D0 00072		MOVL	RO, STATUS	
				56	E8 00075		BLBS	STATUS, 7\$	
				50	D0 00078	6\$:	MOVL	STATUS, RO	: 0646
C3F4	C7	E1F4	C7	54	29 0007C	7\$:	CMPC3	CURRENT_SIZE, GROUP_TEST_DATA, DISK_TEXT	: 0653
				06	13 00084		BEQL	8\$	
		50	01F4	8F	3C 00086		MOVZWL	#500, RO	: 0655
				04	0008B		RET		
	B8		55	03	F3 0008C	8\$:	AOBLEQ	#3, TEST_INDEX, 5\$: 0652
			50	01	D0 00090		MOVL	#1, RO	: 0658
				04	00093		RET		: 0660

; Routine Size: 148 bytes, Routine Base: SCODES + 01AF

```
670      0661 1 ROUTINE DO_QIOW(FUNCTION,P1,P2,P3,P4,P5)=  
671      0662 1  
672      0663 1 !++  
673      0664 1 FUNCTIONAL DESCRIPTION:  
674      0665 1  
675      0666 1 COMMON ROUTINE FOR PERFORMING SQIOW SYSTEM SERVICE  
676      0667 1  
677      0668 1 FORMAL PARAMETERS:  
678      0669 1  
679      0670 1  
680      0671 1 FUNCTION: THE QIOW FUNCTION CODE  
681      0672 1 P1: THE ADDRESS OF THE P1 PARAMETER  
682      0673 1 P2: THE ADDRESS OF THE P2 PARAMETER  
683      0674 1 P3: THE ADDRESS OF THE P3 PARAMETER  
684      0675 1 P4: THE ADDRESS OF THE P4 PARAMETER  
685      0676 1 IMPLICIT INPUTS:  
686      0677 1  
687      0678 1 CHANNEL: THE CHANNEL NUMBER TO THE FILES ACP  
688      0679 1 IOSB: THE IO STATUS BLOCK  
689      0680 1  
690      0681 1 IMPLICIT OUTPUTS:  
691      0682 1  
692      0683 1  
693      0684 1  
694      0685 1  
695      0686 1  
696      0687 1  
697      0688 1  
698      0689 1  
699      0690 1  
700      0691 1  
701      0692 1  
702      0693 1  
703      0694 1  
704      0695 1  
705      0696 1  
706      0697 1  
707      0698 1  
708      0699 1  
709      0700 1  
710      0701 1  
711      0702 1  
712      0703 1  
713      P 0704 1  
714      P 0705 1  
715      P 0706 1  
716      P 0707 1  
717      P 0708 1  
718      P 0709 1  
719      P 0710 1  
720      0711 1  
721      0712 1  
722      0713 1  
723      0714 1  
724      0715 1  
725      0716 1  
726      0717 1  
          2 BEGIN LOCAL STATUS:  
          1 * DO QIOW AND CHECK IO SERVICE RETURN  
          1 IF NOT (STATUS=$QIOW(CHAN=.CHANNEL,IOSB=IOSB,  
          1           FUNC=.FUNCTION,  
          1           P1=.P1,  
          1           P2=.P2,  
          1           P3=.P3,  
          1           P4=.P4,  
          1           P5=.P5))  
          1 THEN RETURN .STATUS;  
          1 * CHECK IO COMPLETION RETURN  
          2 IF
```

```

: 727      0718 2      NOT .IOSB[0]
: 728      0719 2      THEN
: 729      0720 2      RETURN .IOSB[0]
: 730      0721 2      ELSE
: 731      0722 2      RETURN TRUE
: 732      0723 2
: 733      0724 1 END:

```

				.EXTRN	SYSSQIOW	
			0004 00000 DO_QIOW:	WORD	Save R2	: 0661
		52 0000	CF 9E 00002	MOVAB	IOSB, R2	
			7E D4 00007	CLRL	-(SP)	
		7E 14	AC 7D 00009	MOVQ	P4, -(SP)	: 0710
		7E 0C	AC 7D 0000D	MOVQ	P2, -(SP)	
		08	AC DD 00011	PUSHL	P1	
			7E 7C 00014	CLRQ	-(SP)	
			52 DD 00016	PUSHL	R2	
		7E 04	AC DD 00018	PUSHL	FUNCTION	
			CF 3C 0001B	MOVZWL	CHANNEL, -(SP)	
		0000000G	7E D4 00020	CLRL	-(SP)	
		00	OC FB 00022	CALLS	#12, SYSSQIOW	
		0A	50 E9 00029	BLBC	STATUS, 2\$	
		04	62 F8 0002C	BLBS	IOSB, 1\$: 0718
		50	62 3C 0002F	MOVZWL	IOSB, R0	: 0722
			04 00032	RET		
		50	01 D0 00033	MOVL	#1, R0	
			1\$: 04 00036	RET		: 0724

: Routine Size: 55 bytes, Routine Base: \$CODE\$ + 0243

```
735      0725 1 GLOBAL ROUTINE DATA_INIT:NOVALUE=
736      0726 1
737      0727 1
738      0728 1
739      0729 1
740      0730 1
741      0731 1
742      0732 1
743      0733 1
744      0734 1
745      0735 1
746      0736 1
747      0737 1
748      0738 1
749      0739 1
750      0740 1
751      0741 1
752      0742 1
753      0743 1
754      0744 1
755      0745 1
756      0746 1
757      0747 1
758      0748 1
759      0749 1
760      0750 1
761      0751 1
762      0752 1
763      0753 1
764      0754 1
765      0755 2 BEGIN
766      0756 2 REGISTER
767      0757 2   POINTER,
768      0758 2   END_POINTER;
769      0759 2 LITERAL
770      0760 2   WORST_CASE_PAT=%0'165555'^16+%0'133333';
771      0761 2
772      0762 2
773      0763 2 !INIT POINTERS TO BUFFER
774      0764 2
775      0765 2   POINTER=GROUP_TEST_DATA[0];
776      0766 2   END_POINTER=(GROUP_SIZE^4)+.POINTER;
777      0767 2
778      0768 2
779      0769 2 !FILL BUFFER WITH WORST CASE PATTERN
780      0770 2
781      0771 2   WHILE .POINTER NEQU .END_POINTER DO
782      0772 2     BEGIN
783      0773 2       .POINTER=WORST_CASE_PAT;
784      0774 2       .POINTER=.POINTER+4
785      0775 2     END;
786      0776 2
787      0777 2   RETURN
788      0778 2
789      0779 2
790      0780 1   END;
```

50	0000'	0000 0000	.ENTRY	DATA INIT, Save nothing
51	01E0	CF 9E 0002	MOVAB	GROUP TEST DATA, POINTER
51		CO 9E 0007	MOVAB	480(R0), END_POINTER
		50 D1 0000C 1\$:	CMPL	POINTER, END_POINTER
		09 13 0000F	BEQL	2\$
80	EB6DB6DB	8F D0 00011	MOVL	#-345131301, (POINTER)+
		F2 11 00018	BRB	1\$
		04 0001A 2\$:	RET	

: 0725
: 0765
: 0766
: 0771
: 0773
: 0774
: 0780

: Routine Size: 27 bytes, Routine Base: \$CODE\$ + 027A

```
792 0781 1 ROUTINE CHECK_BADSTATUS(STATUS)=  
793 0782 1  
794 0783 1 ++  
795 0784 1 FUNCTIONAL DESCRIPTION:  
796 0785 1  
797 0786 1 ROUTINE CLASSIFYS THE SYSTEM SERVICE CODES THAT IT RECEIVES  
798 0787 1 AS INPUT INTO 3 CATEGORIES  
799 0788 1 NORMAL_STS: $$$ NORMAL  
800 0789 1 BAD_STS: DEVICE ERROR INDICATING A BAD BLOCK  
801 0790 1 ERROR_STS: UNRECOVERABLE DEVICE ERROR  
802 0791 1  
803 0792 1 FORMAL PARAMETERS:  
804 0793 1  
805 0794 1 STATUS: A SYSTEM SERVICE CODE  
806 0795 1  
807 0796 1 IMPLICIT INPUTS:  
808 0797 1  
809 0798 1  
810 0799 1  
811 0800 1 IMPLICIT OUTPUTS:  
812 0801 1  
813 0802 1  
814 0803 1  
815 0804 1  
816 0805 1 ROUTINE VALUE:  
817 0806 1 COMPLETION CODES:  
818 0807 1 RETURNS AS A VALUE ON OF THE 3 ABOVE MENTIONED CODES  
819 0808 1  
820 0809 1  
821 0810 1  
822 0811 1 SIDE EFFECTS:  
823 0812 1  
824 0813 1  
825 0814 1  
826 0815 1  
827 0816 2 BEGIN  
828 0817 2  
829 0818 2  
830 0819 2 ! POSSIBLE IO CODES ARE DIVIDED INTO THREE CASES  
831 0820 2 ! GOOD BLOCKS,BAD BLOCKS AND SEVERE DEVICE ERRORS  
832 0821 2  
833 0822 2 SELECTONE .STATUS OF  
834 0823 2 SET  
835 0824 2 [$$$_NORMAL]:RETURN NORMAL_STS;  
836 0825 2 [$$$_PARITY,  
837 0826 2 $$$_CTRLERR,  
838 0827 2 $$$_DRVERR]: RETURN BAD_STS;  
839 0828 2 [OTHERWISE]:RETURN ERROR_STS  
840 0829 2 TES:  
841 0830 1 END;
```

0000 00000 CHECK_BADSTATUS:
.WORD Save nothing

: 0781

	50	04	AC	D0	00002		MOVL	STATUS, R0		0822
	01		50	D1	00006		CMPL	R0, #1		0824
			03	12	00009		BNEQ	1\$		
			50	D4	0000B		CLRL	R0		
				04	0000D		RET			
00000054	BF		50	D1	0000E	1\$:	CMPL	R0, #84		0825
0000008C	BF		12	13	00015		BEQL	2\$		
000001F4	BF		50	D1	00017		CMPL	R0, #140		
			09	13	0001E		BEQL	2\$		
			50	D1	00020		CMPL	R0, #500		
			04	12	00027		BNEQ	3\$		
	50		02	D0	00029	2\$:	MOVL	#2, R0		0827
			04	0002C			RET			
	50		01	D0	0002D	3\$:	MOVL	#1, R0		0828
			04	00030			RET			0830

: Routine Size: 49 bytes, Routine Base: \$CODE\$ + 0295

```
843 0831 1 ROUTINE NORMAL_COMPLETE:NOVALUE=
844 0832 1
845 0833 1 ++
846 0834 1 FUNCTIONAL DESCRIPTION:
847 0835 1
848 0836 1 CALLED AFTER ENTIRE FILE HAS BEEN SCANNED FOR BAD BLOCKS
849 0837 1 ANY OF THE FILE REMAINING IS GOOD AND SHOULD BE
850 0838 1 RETURNED TO THE VOLUME. FILE IS DELETED AND DEACCESSED
851 0839 1
852 0840 1 FORMAL PARAMETERS:
853 0841 1
854 0842 1 NONE
855 0843 1
856 0844 1 IMPLICIT INPUTS:
857 0845 1
858 0846 1 FIB: FILE IDENTIFICATION OF CURRENT FILE
859 0847 1
860 0848 1 IMPLICIT OUTPUTS:
861 0849 1
862 0850 1 NONE
863 0851 1
864 0852 1 ROUTINE VALUE:
865 0853 1 COMPLETION CODES:
866 0854 1
867 0855 1 NONE
868 0856 1
869 0857 1 SIDE EFFECTS:
870 0858 1
871 0859 1 NONE
872 0860 1
873 0861 1 !--
874 0862 1
875 0863 2 BEGIN
876 0864 2 LOCAL
877 0865 2 STATUS:
878 0866 2
879 0867 2
880 0868 2 !TRUNCATE ANY OF THE FILE THAT REMAINS
881 0869 2
882 0870 2 STATUS=TRUNCATE(1);
883 0871 2 IF
884 0872 2 (.STATUS NEQ SSS_NORMAL) AND
885 0873 2 (.STATUS NEQ SSS_ENDOFFILE)
886 0874 2 THEN
887 0875 2 BEGIN
888 0876 2 ERROR_COMPLETE();
889 0877 2 RETURN
890 0878 2 END;
891 0879 2
892 0880 2
893 0881 2 !DELETE THE FILE
894 0882 2
895 0883 2 IF
896 0884 2 NOT DO_QIOW(IOS_DELETE+IOSM_DELETE,FIB_DESC,0,0,0,0)
897 0885 2 THEN
898 0886 2 BEGIN
899 0887 2 ERROR_COMPLETE();
```

```

900 0888    RETURN
901 0889    END;
902 0890
903 0891    !*
904 0892    !DEACCESS THE FILE
905 0893
906 0894    DO_QIOW(IOS_DEACCESS,FIB_DESC,0,0,0,0);
907 0895
908 0896    RETURN
909 0897    END;

```

0000 00000 NORMAL_COMPLETE:

				WORD	Save nothing	0831
				PUSHL	#1	0870
FDDA	CF	01	01 DD 00002	CALLS	#1, TRUNCATE	0872
			01 FB 00004	CMPL	STATUS, #1	0873
00000870	8F	09	50 D1 00009	BEQL	1\$	0884
		50	D1 0000C	CMPL	STATUS, #2160	0885
		15	12 00015	BNEQ	2\$	0886
		7E	7C 00017	1\$: CLRQ	-(SP)	0887
		7E	7C 00019	CLRQ	-(SP)	0888
		0000'	CF 9F 0001B	PUSHAB	FIB_DESC	0889
FF54	CF	0135	8F 3C 0001F	MOVZWL	#309, -(SP)	0890
		06	FB 00024	CALLS	#6, DO_QIOW	0891
0000V	CF	50	E8 00029	BLBS	R0, 3\$	0892
		00	FB 0002C	CALLS	#0, ERROR_COMPLETE	0893
		04	00031	RET		0894
		7E	7C 00032	3\$: CLRQ	-(SP)	0895
		7E	7C 00034	CLRQ	-(SP)	0896
		0000'	CF 9F 00036	PUSHAB	FIB_DESC	0897
		34	DD 0003A	PUSHL	#52	
FF3C	CF	06	FB 0003C	CALLS	#6, DO_QIOW	
		04	00041	RET		

; Routine Size: 66 bytes. Routine Base: \$CODE\$ + 02C6

```

911 0898 1 ROUTINE ERROR_COMPLETE:NOVALUE=
912 0899 1
913 0900 1 ++
914 0901 1 FUNCTIONAL DESCRIPTION:
915 0902 1
916 0903 1 CALLED WHEN A FATAL DEVICE ERROR OR SYSTEM SERVICE ERROR
917 0904 1 IS ENCOUNTERED DURING PROCESSING. THE CURRENT FILE IS DEACCESSED
918 0905 1
919 0906 1 FORMAL PARAMETERS:
920 0907 1
921 0908 1      NONE
922 0909 1
923 0910 1 IMPLICIT INPUTS:
924 0911 1
925 0912 1      NONE
926 0913 1
927 0914 1 IMPLICIT OUTPUTS:
928 0915 1
929 0916 1      NONE
930 0917 1
931 0918 1 ROUTINE VALUE:
932 0919 1 COMPLETION CODES:
933 0920 1
934 0921 1      NONE
935 0922 1
936 0923 1 SIDE EFFECTS:
937 0924 1
938 0925 1      NONE
939 0926 1
940 0927 1 --
941 0928 1
942 0929 2 BEGIN
943 0930 2
944 0931 2 /*
945 0932 2 !DEACCESS THE FILE
946 0933 2
947 0934 2 DO_QIOW(IOS_DEACCESS,FIB_DESC,0,0,0,0);
948 0935 2
949 0936 2
950 0937 2 RETURN
951 0938 1 END;

```

0000 00000 ERROR_COMPLETE:

			.WORD	Save nothing
	7E	7C 00002	CLRQ	-(SP)
	7E	7C 00004	CLRQ	-(SP)
0000'	CF	9F 00006	PUSHAB	FIB_DESC
	34	DD 0000A	PUSHL	#52
FF2A CF	06	FB 0000C	CALLS	#6, DO_QIOW
	04	00011	RET	

: Routine Size: 18 bytes, Routine Base: \$CODE\$ + 0308

0898
0934

0938

SCANFILE
V04-000

I 1
15-Sep-1984 23:36:57
14-Sep-1984 11:54:33 VAX-11 Bliss-32 V4.0-742
[BADBLK.SRC]SCANFILE.B32;1

Page 30
(14)

```
953 0939 1 ROUTINE GROUP_RETURN=
954 0940 1 ++
955 0941 1 FUNCTIONAL DESCRIPTION:
956 0942 1
957 0943 1
958 0944 1 CALLED WHEN A BAD BLOCK ERROR IS ENCOUNTERED BY
959 0945 1 GROUP BLOCK TESTING. THE INDIVIDUAL BLOCKS IN A GROUP
960 0946 1 ARE TESTED FOR 'BADNESS' AND TRUNCATED OFF THE CURRENT
961 0947 1 FILE AND INTO THE BAD BLOCK FILE WHEN FOUND
962 0948 1 FORMAL PARAMETERS:
963 0949 1
964 0950 1 NONE
965 0951 1
966 0952 1 IMPLICIT INPUTS:
967 0953 1
968 0954 1 STARTING_BLOCK: FIRST BLOCK IN GROUP
969 0955 1 LAST_BLOCK: LAST BLOCK IN GROUP
970 0956 1
971 0957 1 IMPLICIT OUTPUTS:
972 0958 1
973 0959 1 NONE
974 0960 1
975 0961 1 ROUTINE VALUE:
976 0962 1 COMPLETION CODES:
977 0963 1
978 0964 1 NONE
979 0965 1
980 0966 1 SIDE EFFECTS:
981 0967 1
982 0968 1 NONE
983 0969 1
984 0970 1 !--
985 0971 1
986 0972 2 BEGIN
987 0973 2 LOCAL
988 0974 2   VBN:
989 0975 2
990 0976 2
991 0977 2 !INDIVIDUALLY CONSIDER ALL BLOCKS IN THE GROUP
992 0978 2 !RETURN EACH TO THE BADBLOCK FILE OR FREE SPACE
993 0979 2
994 0980 2
995 0981 2 VBN=.LAST_BLOCK;
996 0982 2 WHILE TRUE DO
997 0983 2   BEGIN
998 0984 2     CASE CHECK_BADSTATUS(BLOCKTEST(.VBN))
999 0985 2       FROM NORMAL_STS TO BAD_STS OF
1000 0986 2       SET
1001 0987 2
1002 0988 2     [NORMAL_STS]:TRUNC_BLOCK=.VBN;
1003 0989 2
1004 0990 2     [ERROR_STS]:RETURN FALSE;
1005 0991 2
1006 0992 2     [BAD_STS]:TRUNCATE_BAD(.VBN);
1007 0993 2
1008 0994 2   TES:
1009 0995 2   VBN=.TRUNC_BLOCK-1;
```

```

: 1010      0996
: 1011      0997    IF
: 1012      0998    THEN .VBN LSS .STARTING_BLOCK
: 1013      0999    RETURN TRUE
: 1014          END
: 1015      1000
: 1016      1001
: 1016      1002 1 END;

```

000C 00000 GROUP_RETURN:								
								: 0939
	53	0000' 0C	CF A3	9E DD 00007	1\$:	MOVAB	Save R2,R3	
	52			52 01	0000B FB	MOVL	TRUNC_BLOCK. R3	
	FDFC	CF		50 50	0000D 00012	PUSHL	LAST_BLOCK, VBN	: 0980
	FF62	CF		01 50	00014 CF	CALLS	#1, BLOCKTEST	: 0984
02	00	000B		0006	00019 0001D	PUSHL	R0	
000E					2\$: 0001D	CALLS	#1, CHECK_BADSTATUS	
						CASEL	R0, #0, #2	
						.WORD	3\$-2\$,-	
							4\$-2\$,-	
							5\$-2\$	
	63			52 0A	00023 11	MOVL	VBN, TRUNC_BLOCK	: 0988
				50 01	00026 FB	BRB	6\$	
				50 04	00028 0002A	CLRL	R0	: 0990
				52 01	0002B 0002D	RET		
	52	FD8B	CF	52 01	0002D FB	PUSHL	VBN	: 0992
		63	08	52 01	00032 C3	CALLS	#1, TRUNCATE_BAD	
		A3		52 52	00032 D1	SUBL3	#1, TRUNC_BLOCK, VBN	: 0995
				CF 01	00036 18	CMPL	VBN, STARTING_BLOCK	: 0997
				50 04	0003A 0003C	BGEQ	1\$	
				01 01	0003C 0003F	MOVL	#1, R0	: 0999
						RET		: 1002

: Routine Size: 64 bytes, Routine Base: \$CODE\$ + 031A

SCANFILE
V04-000

L 1
15-Sep-1984 23:36:57
14-Sep-1984 11:54:33 VAX-11 Bliss-32 V4.0-742
[BADBLK.SRC]SCANFILE.B32;1

Page 33
(18)

: 1020 1003 1 END
: 1021 1004 0 ELUDOM

!End of module

PSECT SUMMARY

Name	Bytes	Attributes
\$OWNS	15476	NOVEC, WRT, RD ,NOEXE,NOSHR, LCL. REL. CON,NOPIC,ALIGN(2)
\$SPLITS	8	NOVEC,NOWRT, RD ,NOEXE,NOSHR, LCL. REL. CON,NOPIC,ALIGN(2)
\$CODES	858	NOVEC,NOWRT, RD , EXE,NOSHR, LCL. REL. CON,NOPIC,ALIGN(2)

Library Statistics

File	Symbols			Pages Mapped	Processing Time
	Total	Loaded	Percent		
\$_\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	27	0	1000	00:01.8

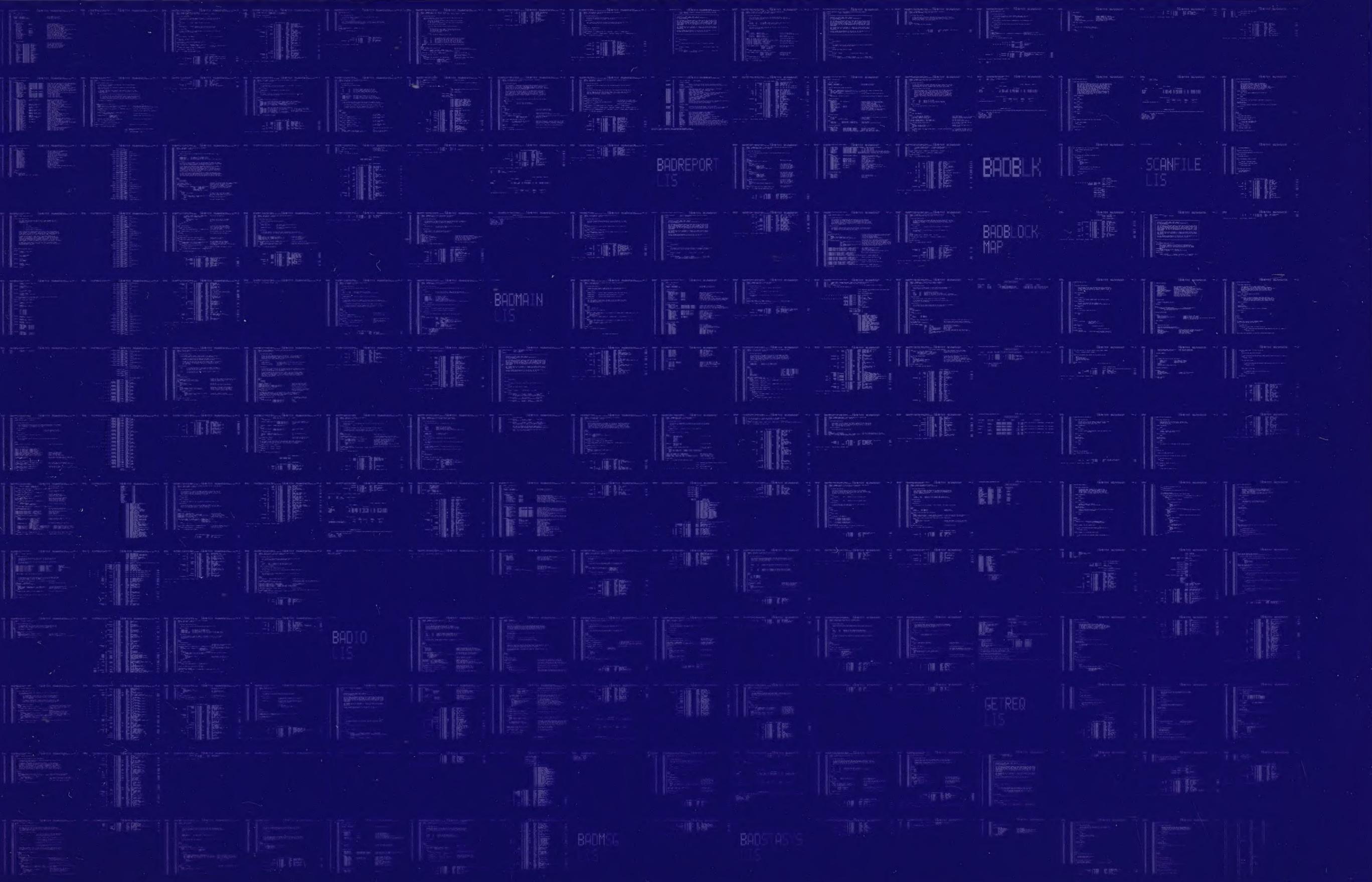
COMMAND QUALIFIERS

: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:SCANFILE/OBJ=OBJ\$:SCANFILE MSRC\$:SCANFILE/UPDATE=(ENH\$:SCANFILE)

: Size: 858 code + 15484 data bytes
: Run Time: 00:17.7
: Elapsed Time: 00:33.5
: Lines/CPU Min: 3407
: Lexemes/CPU-Min: 9369
: Memory Used: 91 pages
: Compilation Complete

0018 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



0019 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY